Journal of Computational Science 83 (2024) 102426

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

DeepDetect: An innovative hybrid deep learning framework for anomaly detection in IoT networks

Zeenat Zulfiqar [a], Saif U.R. Malik [b], Syed Atif Moqurrab [c], Zubair Zulfiqar [d], Usman Yaseen [e], Gautam Srivastava [f,g,h,i,*]

[a] Department of Computer Science, COMSATS University, Islamabad, Pakistan
[b] Information Security Institute, Cybernetica AS, Estonia
[c] School of Computer Science and Technology, University of Bedfordshire, University Square, Luton, LU1 3JU, United Kingdom
[d] Department of Software Engineering, National University of Sciences and Technology (NUST), Pakistan
[e] University of Derby, United Kingdom
[f] Department of Math and Computer Science, Brandon University, Canada
[g] Department of Computer Science and Math, Lebanese American University, Lebanon
[h] Research Centre for Interneural Computing, China Medical University, Taichung, Taiwan
[i] Centre for Research Impact & Outcome, Chitkara University Institute of Engineering and Technology, Chitkara University, Rajpura, 140401, Punjab, India

ARTICLE INFO

Keywords:
Anomaly detection
Network security
5G
Deep learning
Internet of things
NSL-KDD

ABSTRACT

The presence of threats and anomalies in the Internet of Things infrastructure is a rising concern. Attacks, such as Denial of Service, User to Root, Probing, and Malicious operations can lead to the failure of an Internet of Things system. Traditional machine learning methods rely entirely on feature engineering availability to determine which data features will be considered by the model and contribute to its training and classification and "dimensionality" reduction techniques to find the most optimal correlation between data points that influence the outcome. The performance of the model mostly depends on the features that are used. This reliance on feature engineering and its effects on the model performance has been demonstrated from the perspective of the Internet of Things intrusion detection system. Unfortunately, given the risks associated with the Internet of Things intrusion, feature selection considerations are quite complicated due to the subjective complexity. Each feature has its benefits and drawbacks depending on which features are selected. Deep structured learning is a subcategory of machine learning. It realizes features inevitably out of raw data as it has a deep structure that contains multiple hidden layers. However, deep learning models such as recurrent neural networks can capture arbitrary-length dependencies, which are difficult to handle and train. However, it is suffering from exploiting and vanishing gradient problems. On the other hand, the log-cosh conditional variational Autoencoder ignores the detection of the multiple class classification problem, and it has a high level of false alarms and a not high detection accuracy. Moreover, the Autoencoder ignores to detect multi-class classification. Furthermore, there is evidence that a single convolutional neural network cannot fully exploit the rich information in network traffic. To deal with the challenges, this research proposed a novel approach for network anomaly detection. The proposed model consists of multiple convolutional neural networks, gate-recurrent units, and a bi-directional-long-short-term memory network. The proposed model employs multiple convolution neural networks to grasp spatial features from the spatial dimension through network traffic. Furthermore, gate recurrent units overwhelm the problem of gradient disappearing- and effectively capture the correlation between the features. In addition, the bi-directional-long short-term memory network approach was used. This layer benefits from preserving the historical context for a long time and extracting temporal features from backward and forward network traffic data. The proposed hybrid model improves network traffic's accuracy and detection rate while lowering the false positive rate. The proposed model is evaluated and tested on the intrusion detection benchmark NSL-KDD dataset. Our proposed model outperforms other methods, as evidenced by the experimental results. The overall accuracy of the proposed model for multi-class classification is 99.31% and binary-class classification is 99.12%.

* Corresponding author at: Department of Math and Computer Science, Brandon University, Canada.
E-mail addresses: zeenatzulfiqar95@gmail.com (Z. Zulfiqar), aif.rehmanmalik@cyber.ee (S.U.R. Malik), syedatif.moqurrab@beds.ac.uk (S.A. Moqurrab), zubairzulfiqar96@gmail.com (Z. Zulfiqar), sman.yaseen@comsats.edu.pk (U. Yaseen), srivastavag@brandonu.ca (G. Srivastava).

https://doi.org/10.1016/j.jocs.2024.102426
Received 26 November 2023; Received in revised form 9 July 2024; Accepted 20 August 2024
Available online 6 September 2024

1. Introduction

With the advent of the virtual world, the Internet has become deeply integrated into our daily lives. The use of Internet of Things (IoT) technology, such as 5G, smart cities, cloud computing, self-driving vehicles, finance, health monitoring via smartwatches, and cyberspace, has been adopted by businesses and governments for critical services and daily operations [1]. It is projected that by 2030, there will be 500 billion connected devices worldwide [2]. As a result, the traffic volume of IoT devices is increasing rapidly. However, the current 5G infrastructure cannot handle this growing traffic volume. To address the limitations of 5G, the upcoming 6G technology aims to establish a new radio environment, but it also brings specific vulnerabilities [3].

The IoT generates massive amounts of data from various applications, enabling communication between multiple sensors and devices without human interaction. To facilitate the gathering, communication, storage, retrieval, exchange, and transmission of real-world data, a platform is required to connect all applications, goods, and services [4]. However, these advancements also introduce significant network security risks, as evidenced by the increasing number of network breaches and cybercrime incidents [5].

Addressing security concerns on the Internet is imperative to analyze the patterns of malicious activities and develop flexible strategies to meet network security requirements [6]. Traditional encryption techniques, firewalls, network security protocols, and similar tools are commonly used by organizations to protect against network threats. However, they often struggle to detect new and complex threats [7]. Network intrusion detection systems (NIDS) that utilize network signatures have emerged as a defensive measure, but they are limited in identifying new or unexpected threats [8]. To mitigate attacks and terrorist risks to IoT data and infrastructure, new machine learning (ML)-based solutions are being developed [9].

ML techniques are well-suited for IoT networks compared to traditional signature-based approaches as they do not rely on extensive databases for signature matching [9]. The traffic in a network can broadly be classified as malicious and benign traffic, representing a classification problem [10]. While ML approaches like Support Vector Machine (SVM), Naive Bayes (NB), Decision Tree (DT), or Random Forest (RF) have been employed to build network intrusion detection (ID) classifiers, they often suffer from high false positive rates, low detection and accuracy rates, and reliance on manual traffic design features, which become less effective with large volumes or imbalanced data [11].

To overcome the limitations of traditional ML ID methods, Deep Learning (DL)-based techniques have been introduced. As a more advanced branch of ML, DL can overcome some of the weaknesses of shallow learning [12]. Various DL models such as Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Multiple layer RNN, LCCV-AE, and AE have been utilized to detect anomalies in network traffic data [6,13,14]. However, these DL approaches have their limitations, such as gradient vanishing issues and low detection rates (DR), and they may fail to detect multi-class classification [13,14].

This research introduces DeepDetect, a hybrid DL model composed of multiple layers of CNN-GRU-Bi-LSTM, to analyze abnormalities in network traffic data. The model combines DL models with a multi-layer CNN approach for spatial feature extraction, gated recurrent units to capture feature correlation, and Bi-LSTM for temporal feature extraction from network traffic data. The proposed model aims to enhance the accuracy of network traffic data, reduce false positive rates, and improve the DR for each class label. The model's performance is evaluated via extensive experiments using the NSL-KDD benchmark dataset. The results are compared against the current state-of-the-art DL techniques.

The main contributions of our research are as follows:

- The Unification of deep learning models, specifically CNN, GRU, and Bi-LSTM, significantly enhances the accuracy of network traffic data analysis. This combination not only reduces the false positive rate (FPR) but also boosts the detection rate (DR) for all class labels.
- Evaluating the performance of the proposed model both on multi-class and binary-class classification.
- A multi-layer Convolutional Neural Network (CNN) approach is utilized to extract and learn spatial features from the spatial dimensions of network traffic data. This technique allows the model to identify and understand complex patterns and structures within the spatial dimensions, thereby improving its accuracy in network traffic analysis.
- Solving the gradient vanishing problem and effectively capturing the correlation between the features through GRU.
- The Bi-LSTM approach is used to maintain historical content for a long time and can extract temporal features from both backward and forward network traffic data.
- Finally, evaluate the proposed model on the NSL-KDD benchmark dataset and compare it with current state-of-the-art deep learning approaches.

The remainder of this document is organized as follows. The Related Work is covered in Section 2. Material and Methods are included in Section 3. Section 4 contains the proposed model. Section 5 shows the findings of the experiment setting. The results of the Proposed Model are discussed in Section 6. Comparison with the SOTA Model for NSL-KDD is shown in Section 7. The conclusion is found in Section 8.

2. Related work

This section summarizes existing studies on detecting network anomalies in IoT data environments using ML and DL approaches also the motivation of research.

ML models employ feature extraction techniques, while DL models integrate novel feature engineering methods. Both ML and DL methods are extensively utilized for anomaly detection in IoT security. Attacks on IoT infrastructure, including Data Type Probing (DTP), Malicious Operation (MO), Malicious Control (MC), Denial of Service (DoS), scan, Wrong Setup (WS), and spying, pose risks of system failures [15]. Traditional signature-based intrusion detection systems (IDS) require significant storage and processing time for pattern matching. To address this, a GPU-based MapReduce approach achieved a four-fold speed improvement compared to CPU implementations [16]. However, signature-based IDSs struggle with novel or unforeseen threats. In conventional environments, ML methods have been widely employed for network intrusion detection (ID) [15].

One study employed various ML models, achieving reliable accuracy using Random Forest (RF) and Artificial Neural Networks (ANN) for detecting attacks on IoT systems. However, the authors did not test the algorithms on benchmark datasets, and the effectiveness of RF with large data or unknown issues remains uncertain. Another approach used the K-Nearest Neighbor (KNN) technique and a two-step hybrid method for binary classification [17]. However, this study lacked accuracy evaluation and did not report the recall rate.

In Ref. [18], a novel approach for ID utilized Extreme Learning Machine (ELM), Support Vector Machine (SVM), and RF techniques. Although this achieved high accuracy, it suffered from a high False Positive Rate (FPR) and overfitting. An adaptive algorithm was suggested in [19] to improve overall accuracy using RF, KNN, Decision Tree (DT), and Deep Neural Networks (DNN). However, the Detection Rate (DR) for attacks classified as User-to-Root (U2R) and Remote-to-Local (R2L) was still low. Another lightweight attack detection approach based on an SVM classifier used only three features derived from packet arrival attributes for training [20]. Despite the limited features, this method effectively identified intrusions in IoT networks but had a significantly higher False Acceptance Rate (FAR).

In Software-Defined Networking (SDN), ML algorithms were developed for anomaly detection from network traffic [21]. However, results showed low accuracy when only normal data was available, and developing discriminative feature techniques was challenging.

DL has gained attention, particularly in ID applications, showing superior performance to ML algorithms. DL methods extract valuable insights from vast data volumes and can be categorized into supervised learning (SL) approaches like Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN), as well as unsupervised learning (UN-SL) techniques such as Autoencoders (AE) and Deep Belief Networks (DBN).

To address ML limitations, a Gated Recurrent Unit (GRU) model aimed to understand interconnections in security threat sequence data and generate future alerts based on historical alarms from attacking sources [22]. CNN models have been widely used for anomaly detection, but the CNN-IDS model showed a decline in DR for U2R and R2L attacks [23]. A hybrid Deep Learning model achieved high accuracy but faced challenges in parameter convergence [24]. An adaptive synthetic sampling approach addressed inter-channel information redundancy and unbalanced data distribution in CNN-based ID models but had limitations in detecting R2L and U2R attacks [25].

Classifying attack traffic is crucial in network-based security. A traffic classification system (TCS) based on the Long Short-Term Memory (LSTM) model improved detection by eliminating the need to preprocess packets into flows [26]. The RNN model outperformed traditional methods like ANN and SVM but faced challenges in training time [13]. Converting raw traffic vectors into an image data format improved detection performance for imbalanced traffic datasets, but using a single CNN model resulted in low accuracy [10]. A Bi-LSTM DL model improved accuracy but had high time complexity [27]. An anomaly detection method called LCV-AE utilized the Conditional Variational Autoencoder (CVAE) but suffered from low DR and high FPR [14]. An AE model increased ID accuracy for binary-class classification but requires improvements for multiple-class classification [6].

Extracting different-level features from network connections allowed for more efficient feature processing [28]. A multi-modal sequential ID approach based on a hierarchical progressive network supported by Multidimensional Autoencoder (MDAE) and LSTM technologies required improvements in accuracy and addressing the detection of FAR [28]. The SAE attention mechanism combined with DNN improved network identification but exhibited a relatively low ID and high false positives [29]. Table 1 shows a review matrix of all literature review papers related to machine learning. This table summed up all of the literature review papers. The research description, methodology, and limitations are all included in the tables.

2.1. Motivation of research

In the last few years, anomaly detection has gotten a lot of attention. Emerging technologies, such as the IoT, are known to be among the most important sources of data streams, which continuously produce massive amounts of data from a variety of applications [30]. Anomaly detection identifies patterns that do not match the expected normal patterns [31], which are referred to as anomalies or outliers.

Many researchers have been working on IoT security and network security. However, no new countermeasures have been developed to address constantly updated network detection techniques. In difficult attacks, for example, the attackers will send very specific, low-intensity traffic to the target system that is specific to the application Because they do not use intensity or volume to disrupt network service, these data packets are very similar to legitimate traffic and will not cause significant changes in numerical statistics [32]. Traditional detection and protection techniques, including cryptography, access control, and firewalls, have limitations in providing complete network/security protection against increasingly sophisticated attacks, such as DOS. Furthermore, the majority of systems that rely on these approaches have a high FPR and cannot adapt to malicious behavioral change.

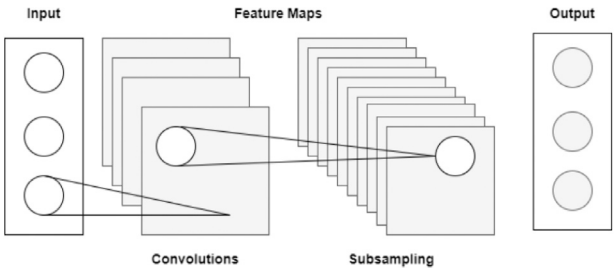


Fig. 1. Architecture of CNN.

ML has been extensively researched for detecting network anomalies [33]. However, our preliminary experiments revealed that when analyzing network data, which is frequently complex and high-dimensional, conventional ML is largely limited, with sometimes unacceptable accuracy in detection. The DL method has recently been used in network intrusion detection systems because it has been shown to successfully solve a variety of problems in research areas such as text classification, object detection, and image classification. Most of the research using the DL approach in this active area of research focuses on dimensionality reduction [34] and anomalous intruder detection [35]. With good layerwise learning, the DL method shows excellent performance equal to or better than the efficiency of the shallow learning method [36].

3. Material and methods

In this section, we go through the details of the suggested framework's theory

3.1. Theory of the proposed framework

The theory of the suggested framework is thoroughly discussed in this subsection.

3.1.1. Convolutional neural networks

CNN is a DL approach that combines filters as input. It includes a pooling layer and a layer of convolution. The mapping features layers and features extraction layers make up the convolutional layer. The input of each neuron in the feature extraction layer is connected to the local sensory domain of the previous layer, enabling the extraction of local features. The network's computing layers consist of multiple feature mappings in the featured mapping layer. Due to weight sharing among neurons on a feature mapping plane, the network effectively reduces the number of free parameters. Convolution result is calculated statistically by the pooling layer, with average and maximum pooling being the most popular computation algorithms. The architecture of the CNN algorithm is shown in Fig. 1.

3.1.2. Gated recurrent units

The GRU network is like the LSTM network. GRU is a new variant of RNN. An RNN is a feed-forward neural network extended to solve variable-length sequences. RNN, on the other hand, is easily influenced by short-term memory. When a sequence is long, RNNs find it hard to transfer meaningful information from the beginning to the end. It enables RNNs to overlook critical information in a long sequence. RNNs seem to have a serious vanishing gradient problem during backpropagation. The weight of our network has been updated using a gradient. If the gradient falls to zero, our network will stop updating. GRU is built with more persistent memory in mind, which makes it smoother for RNNs to apprehend long-term dependencies. The architecture of GRU is shown in Fig. 2. Which consists of two gates that are reset and update gates. The update gate allows the model to determine

Table 1
Related work: Machine learning and Deep learning.

Study	Description	Approach	Limitation
[15]	Comparison of machine learning models to predict the attack on the Internet of things systems accurately.	SVM, LR, DT, RF, ANN	Not tested and trained the algorithms on benchmark data sets. It does not guarantee that RF will work in the presence of large amounts of data or other unknown problems.
[16]	A pattern-matching approach is executed in parallel using the MapReduce framework.	MapReduce approach	Signature-based IDS detect known threats effectively, but they fall short when identifying new or unexpected threats.
[17]	Proposed the K-NN and a two-step hybrid method that supported binary classification to improve detection performance and reduce bias toward frequent attacks.	BC-kNN	Relatively low accuracy and not assess the recall rate.
[18]	Intrusion detection	ELM, RF, SVM	High FPR, Overfitting
[19]	By combining multiple DT with an AVM, and AEM was developed.	KNN, RF, DNN, DT	The accuracy of D2L and U2R attacks are still low.
[20]	Proposed an innovative lightweight way to malware detection threats in IoT.	SVM	Single SVM still has a significantly higher FAR.
[21]	Detecting attacks in SDN	ML techniques	Low accuracy on ML algorithms.
[22]	Prediction of alerts from a malicious source	GRU	Low prediction rate.
[23]	Feature Reduction and proposed CNN-IDS Model	CNN-IDS	The detection of U2R and R2L is minimal.
[24]	Proposed a Deep Multiscale CNN Based ID Method	CNN-IDS	During the model optimization process, parameters cannot converge to the global optimal value.
[25]	Proposed adaptive synthetic sampling (ADASYN) method	AS-CNN	The accuracy of R2L and U2R attacks is low.
[26]	Malicious Traffic classification	LSTM	High dimensionality.
[13]	Proposed RNN-IDS and compared with j48, SVM, and RF	RNN	Need to save time, GPU accelerated training time. During backpropagation, RNN appear to have a serious vanishing and exploiting gradient problem.
[10]	Proposed model for a Massive Network Using Convolutional Neural Networks.	CNN	A single CNN model cannot fully exploit the rich information in network traffic. Detection accuracy is low need to improve.
[27]	Bi-LSTM model for ID	Bi-LSTM	Higher complexity.
[14]	Recommend a novel intrusion detection DL approach which is called LCVAE.	LCV-AE	The DR is poor, and the FPR is large.
[6]	Proposed an AE model to increase the ID accuracy on binary class	AE	Ignore to improve the DR of multiple-class classification. Still, there is a need to improve the accuracy of ID.
[28]	Proposed multimodal sequential intrusion detection approach	MS-DHPN	Low accuracy and ignore to detect the FAR.
[29]	Proposed the SAAE-DNN ID method, which combines an SAE attention mechanism with DNN	SAAE-DNN	Low detection rate.

how much previous data must be passed along in the future. The reset gate determines how much information from the past should be erased. In the following equation, we computed the reset and update gates.

$$RS_t = \sigma(X_t * U_R + H_{t-1} * W_R) \tag{1}$$

RS_t named reset gate, which is responsible for the network's short-term memory, is represented in Eq. (1). RS_t will have a value ranging from 0 to 1 due to the sigmoid function. The weight matrices for the reset gate are U_R and W_R.

$$Z_t = \sigma(X_t * U_Z + H_{t-1} * W_Z) \tag{2}$$

Similarly, the update gate is shown in Eq. (2). Which is responsible for deciding how much previous information needs to proceed in the future.

$$\hat{H}_t = tanh(X_t * U_g + (RS_t H_{t-1}) * U_g) \tag{3}$$

where $\hat{H}_t$ denotes the candidate's hidden state in Eq. (3). It first takes the previous timestamp $t-1$ input and hidden state, multiplies them by a reset gate output RS_t and then passes all this data to the *tanh* function, which returns the candidate hidden state.

$$HS_t = Z_t H_{t-1} + (1 - Z_t)\hat{H}_t \tag{4}$$

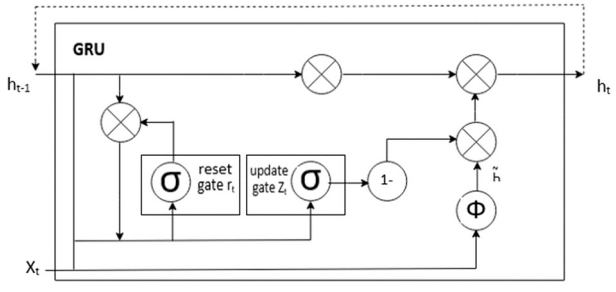


Fig. 2. Gated Recurrent Units Architecture.

Ultimately, the output of the update gate Z_t must be considered. It defines how much of the new candidate's state $\hat{H}_t$ is employed, as well as how much of the new hidden state HS_t This determines how much of the new material will be used. HS_{t-1}. To use update gate Z_t, simply take element-wise convex combinations among HS_{t-1} and $\hat{H}_t$. As a result, the GRU's final update equation is shown in Eq. (4).

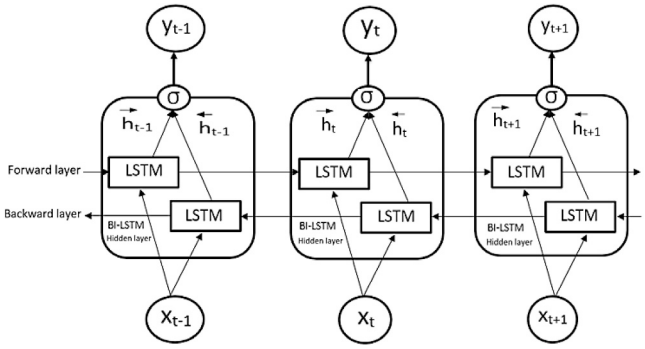


Fig. 3. Architecture of Bi-directional-LSTM.

3.1.3. Bi-directional-LSTM network

A bi-LSTM which is shown in Fig. 3 is a sequential processing model that is split into two LSTMs, one that takes input in one direction and the other that takes input in the opposite direction. Bi-LSTMs are accompanied by a rise in the network's data requirements, providing the algorithm with a better context.

The equations of Bi-LSTM are shown in Eq. (5), Eq. (6), and Eq. (7).

$$\vec{h}s_t = \sigma(W_x \vec{h} X S_t + W S_{\vec{h}\vec{h}} h_{t-1} + b_{\vec{h}}) \tag{5}$$

$$\overleftarrow{h}s_t = \sigma(W_x \overleftarrow{h} X S_t + W S_{\overleftarrow{h}\overleftarrow{h}} h_{t-1} + b_{\overleftarrow{h}}) \tag{6}$$

$$ys_t = W S_{y\vec{h}} \vec{h} + W S_{y\overleftarrow{h}} \overleftarrow{h} + b_y \tag{7}$$

4. DeepDetect: Proposed hybrid model for network anomaly detection

The multiple layers CNN-GRU-Bi-LSTM consist of different layers. Each layer has its function. We map the features into binary vectors to arrange and transform the dataset using the one-hot encoding method. The category features are converted into the n-dimensional binary code vector. After converting it into the numerical form, we apply the normalization process. We employ multiple CNN layers to spatial features from the spatial dimension through network traffic. The batch normalization layers allow parameters between intermediate layers to be normalized, resulting in faster training time. We employ the max-pooling layer, which reduces the over-fitting. GRU overcomes the gradient vanishing problems.

Furthermore, the Bi-LSTM layers can extract temporal features from backward and forward network traffic data and preserve the historical context for a long time. We employ the fully connected dense and SoftMax layers at the output layers. The dense layer performs matrix–vector multiplication. Finally, the SoftMax layers categorize the network traffic. Fig. 4 illustrates the proposed architecture. The proposed model consists of the following stages.

4.1. Preprocessing

We arrange and then convert the NSL-KDD datasets after feeding these into the developed framework.

4.1.1. One-hot encoding

On the dataset, we use the one-hot-encoding approach as there are three symbolic data in NSL-KDD datasets that are protocol type, flag, and service. We map these characteristics into binary vectors using a one-hot encoder. The category features are converted to an n-dimensional binary code vector using the hot encoding method. The total number of attributes for the category features is denoted by "n". TCP, UDP, and ICMP are a type of protocols. The one-hot technique generates a computer-readable binary code with three distinct characteristics: TCP, UDP, and ICMP, each of which is encoded in three three-dimensional binary vectors: [1,0, 0], [0,1,0], and [0,0,1]. In other words, the single "protocol type" feature is split into three via one-hot encoding.

4.1.2. Normalization processing

The process of scaling data over a specific interval to reduce redundancy and improve model training time is known as normalization. To standardize continuous data in the [0, 1] range, we use a standard scaler. Normalization removes the influence of the scale of measurements on model training, increasing the reliance of the model outcomes on data attributes.

Pseudo-code for the proposed model is shown in the below. The proposed model performs anomaly detection in which labels may be binary or multiple. Thus, the output consists of a set of labels.

Algorithm 1: Proposed Model

Input : NSL-KDD Dataset (KDD Train+, KDD Test+)
Output: Accuracy, Precision, Recall, F1-score, FPR, DR

```
Function Preprocess:
    for each data in the training or test set do
        data ← One hot encoding(data)
        data ← Normalize(data)
    end
Function ExtractFeatures:
    model ← Train(KDD Train+ dataset)
    features ← Convolution(model)
    features ← GRU(features)
    features ← Bi-LSTM(features)
    classification ← FullyConnected(features) ;
Function Classify:
    results ← Classify(KDD Test+ dataset)
Return results
```

4.2. Convolutional layers (CONV-1 and CONV-2)

The first two layers are convolutional ID layers (ID-CNN), where these two CNN layers include 64 convolutional filters, 41 features, 3 kernel sizes, and a layer of activation that employs the "Relu" activation function (AF). The convolutional layers use the same padding to ensure the output matches the input. CNN is used because it is well suited for extracting spatial features of network traffic data from large amounts of network data. It also learns spatial features from the spatial dimension, primarily through network traffic.

4.3. Batch normalization layer

To improve the stability and solve the covariate shift issue, we employ one of the most essential supervised learning techniques called Batch Normalization after two layers of CNN. This problem is described as an internal covariate shift. It normalizes the previous layer's inputs; a BN layer is included on top of the max-pooling layer. It allows parameters between intermediate layers to be normalized, resulting in faster training times. The internal covariate shift is a problem in NN. The distribution of data changes as we train our neural network, and the model trains more slowly.

4.4. Max-pooling layer

Pooling of most large pieces from the area of the feature space encompassed by the filter is known as max pooling. The max-pooling layer reduces spatial size and over-fits by abstractly modeling the convolved features. Consequently, the outcome of the max-pooling layer will be a feature map containing the essential features from the previous feature map.

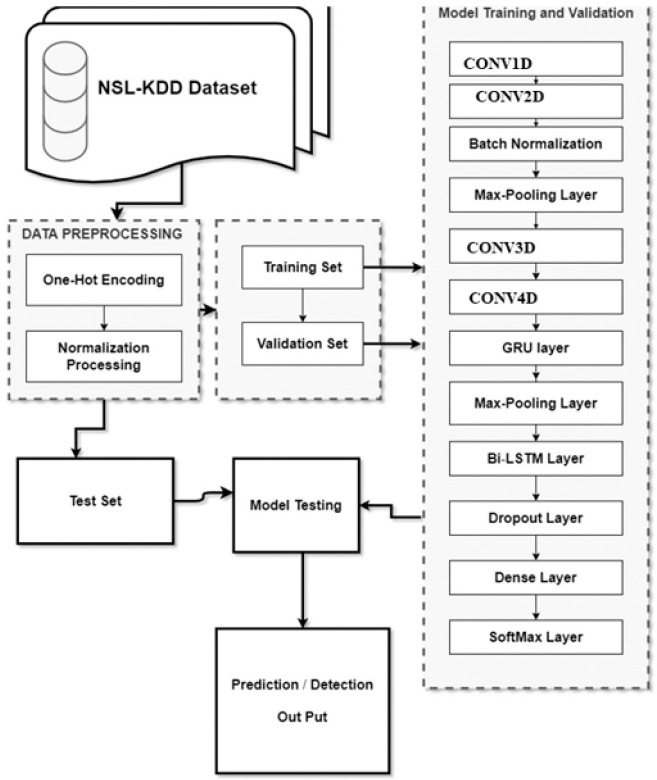


Fig. 4. Proposed system model.

4.5. Convolutional layers (CONV-3 and CONV-4)

Shallower convolutional layers with a narrow receptive field can extract local information, whereas greater layers with a larger vision field can capture global information. As a result, as the number of convolution layers grows, the convolutional feature's scale becomes increasingly coarse. We use two CONV1D layers once again. There have been 128 convolutional filters, three kernel sizes, and an activation layer that employs the "Relu" AF. The convolutional layers use the same padding to ensure that the output is identical to the input.

4.6. Gated recurrent units layer

The GRU layer includes 64 filters and 41 features. We employ the GRU layer after multiple layers of CNN. Using this layer can solve the problem of gradient vanishing and effectively capture the correlation between the features. Although, in general, LSTM and GRU work well with time-series data. On the other hand, the GRU excels at achieving convergence and weight updates. GRU consists of two gates that are reset and update gates. The update gate enables the model to determine how much previous data must be passed along in the future. The reset gate determines how much information from the past should be erased.

4.7. Bi-LSTM layer

The Bi-LSTM layer is the next one we use. This layer has the advantage of maintaining historical context for a long time and can extract temporal features from both backward and forward network traffic data. The Bi-LSTM model, which links the forward and backward LSTM, is employed at the Bi-LSTM layer to extract temporal data from each packet's traffic bytes. Bi-LSTM can acquire the consecutive features of traffic bytes since it is tailored to the structure of network activity.

4.8. Dropout layer

This layer is a probability coefficient that determines how many neurons are discarded during training in a particular layer of a neural network. For example, when 0.1 is used at the dropout layer, 10% of the neurons are dropped from that layer. It helps prevent overfitting.

4.9. Dense layer

We use a fully connected Dense Layer as an output layer. Consequently, in a Neural Network with deep connections, every neuron in the thick layer receives information from all neurons in the layers preceding it. The matrix values are parameters that could be trained and upgraded using backpropagation, and this layer performs mathematical operations. It has the advantage of learning features from all combinational features of the previous layer.

4.10. Soft-max layer

The Softmax layer is utilized to classify network traffic into both binary and multi-class categories. In binary classification, the Softmax layer distinguishes network traffic as Normal or Attack. Similarly, in multi-class classification, the Softmax layer classifies network traffic into five categories: Normal, Dos, URL, R2L, and probe.

5. Experimental setting and discussion

The goal of this section is to conduct experiments on the proposed model and other supplementary models for comparison with the results of the existing paper. The experiments were performed on benchmark NSL-KDD datasets. Further discussion in this section contains the experiment setting, a detailed description of the dataset, implementation details, and performance parameters for the evolutionary model.

5.1. Implementation detail

Python and the Keras library were used to implement multiple-layer CNN-GRU-Bi-LSTM on a PC with a processor Intel(R) Core(TM) i5-2540M CPU @ 2.60 GHz 2.60 GHz microprocessor and 8 GB RAM. The proposed model is implemented using Google Colaboratory, which is an online Jupyter notebook. The suggested model has been trained using the GPU. GPU enables us to execute large datasets quickly. Spyder and Jupyter Notebook were also used in the proposed model. For implementation, the Keras and TensorFlow libraries is used. Keras is a DL API written in Python that runs on the TensorFlow ML framework. It was designed with the goal of allowing for quick experimentation. When conducting research, it is critical to be able to move from concept to result as quickly as possible. TensorFlow is an open-source machine learning platform in its entirety. It is a vast and adaptable ecosystem of tools, libraries, and other resources that provide high-level APIs to processes. The framework offers a variety of principles for creating and deploying machine learning models at various levels. Table 2 shows the hyperparameter setting for proposed model.

5.2. Hyperparameters

In binary-class classification, when the number of classes M equals 2, Binary Cross-Entropy(BCE) can be calculated as:

$$-(y\log(p) + (1-y)\log(1-p)) \tag{8}$$

If $M > 2$ (i.e. multiclass classification), calculate a separate loss for each class label per observation and add the results.

$$-\sum_{c=1}^{M} y_{o,c}\log(p_{o,c}) \tag{9}$$

Table 2
Hyperparameter settings.

Serial No.	Hyperparameter	Value
1	Batch size	30
2	Epoch	50
3	Learning rate	0.00007
4	Optimizer	Adam
5	Activation function	relu
6	Padding	same
7	Kernel	3
8	Cost function	Cross entropy/ binary entropy

Table 3
Dataset.

Dataset	Total	Normal	Dos	Probe	R2L	U2L
KDD Train+	125,973	67,343	45,927	11,656	995	52
KDD Test+	22,544	9711	7458	2421	2754	200
Total dataset	**148,517**	**77,054**	**53,385**	**14,077**	**3,749**	**253**

where M is the number of classes, log is the natural log, y is the binary indicator (0 or 1) if class label c is the correct classification for observation o, and p is the predicted probability observation o is of class c.

The model uses cross-entropy with a 0.00007 learning rate. For the train, the proposed model epoch is set to 50 iterations. Moreover "same" padding is used for all layers. At the output layers, the sigmoid activation function is set for classifications with batch size 30. The sigmoid activation function is depicted in Eq. (11). The Relu activation function is used in each layer, as shown in Eq. (10).

$$Relu(z) = max(0, z) \tag{10}$$

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ i = 1, 2, \ldots, K \tag{11}$$

5.3. Dataset description

This subsection describes the dataset used for training and testing the proposed DeepDetect approach, the NSL-KDD benchmark dataset. It is a variation of the KDD Cup 1999 dataset [37]; it is widely employed. Not only does the NSL-KDD dataset adequately solve the KDD Cup 1999 dataset's inherent redundant record problems. However, it considerably reduces the number of training and testing datasets records. We use KDD Train+ and KDD Test+ as our training and testing datasets, divided into five classes: normal, Dos, R2L, Probing attacks, and U2R attacks, as depicted in Table 3. A data set has 148,517 instants in total, with the KDD Train+ data set having 125,973 records and the KDD Test+ data set having 22,544 records.

5.4. Evaluation matrices

The suggested model uses six evaluation criteria to evaluate the outcome of the baseline and proposed models. The suggested model took advantage of Keras's built-in metrics package. These are as follows.

$$Recall = \frac{TP}{TP + FN} \tag{12}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{13}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{14}$$

$$Precision = \frac{TP}{TP + FP} \tag{15}$$

$$FPR = \frac{FP}{FP + TN} = 1 - \frac{TP}{TP + FN} \tag{16}$$

$$DR = \frac{TP}{TP + FN} \tag{17}$$

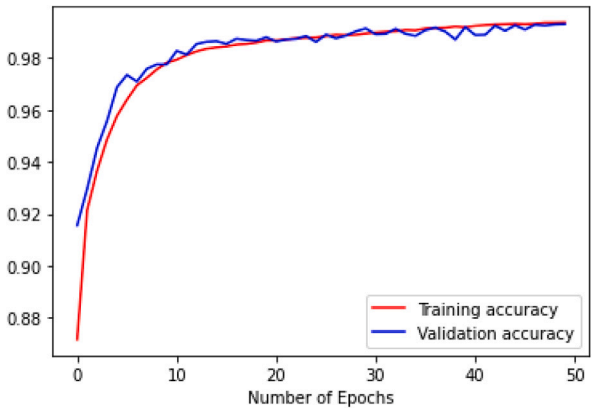


Fig. 5. Training and Validation Accuracy.

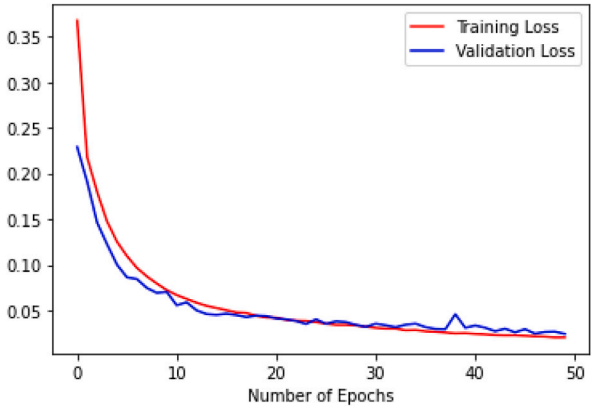


Fig. 6. Training loss and Validation loss.

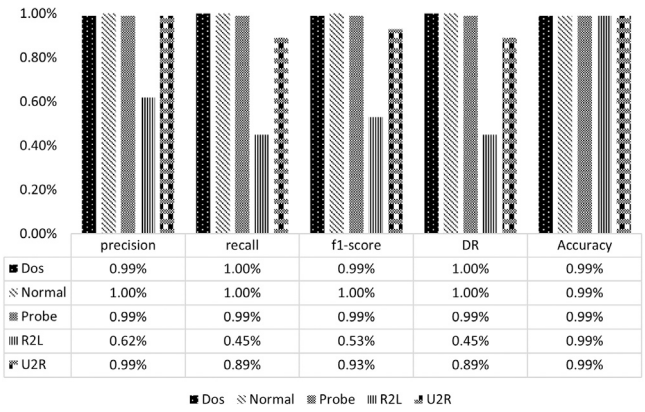
6. Results of proposed model

The success of the multiple-layer DeepDetect models for both binary and multi-class classifications has been investigated through experimental studies. Moreover, for binary classification and multi-classifications, a detailed description is discussed in Sections 6.1 and 6.2.

6.1. Results of multi-class classification

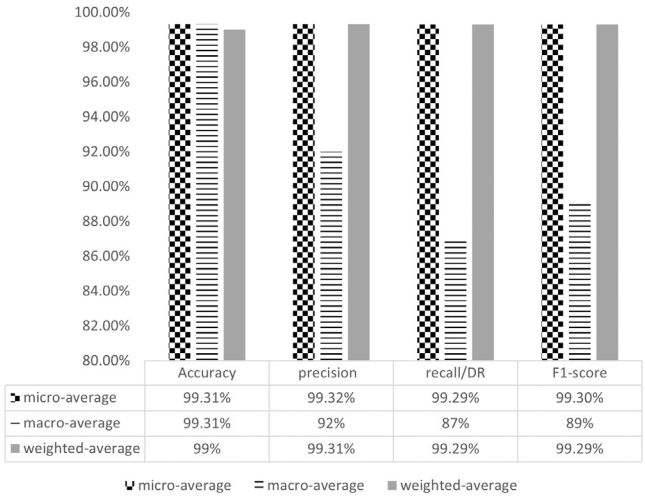
Summarize the thorough explanation of the multi-class categorization results in this subsection. For the multi-class classifications, The model uses cross-entropy with a 0.00007 learning rate. At this learning rate, the model is not facing over-fitting. For the train, the proposed model epoch is set to 50 iterations. Moreover, "same" padding is used for all layers. In multi-class classification with a batch size of 30, the output layers are configured with the sigmoid activation function. The execution time taken by the DeepDetact model for multi-class classification on the NSL-KDD dataset is 1245.37 s, 1245375.324 ms, and 20.7 min. Fig. 5 illustrates the training accuracy and validation graph. The validation accuracy is somewhat raised about the training accuracy, resulting in high accuracy on the training dataset. The number of epochs on the x-axis and training and validation accuracy are set at the y-axis. The validation and training accuracy change with each epoch. The validation and training accuracy for 5 epochs is 92%. For every epoch, the validation and training accuracy are increasing, which is good practice that the model is not facing over-fitting as both validation and training accuracy are slightly increasing upward.

The last epoch ends at 50 because we set the epoch value as 50. The loss of training and loss of validation are depicted in Fig. 6. Here



	precision	recall	f1-score	DR	Accuracy
Dos	0.99%	1.00%	0.99%	1.00%	0.99%
Normal	1.00%	1.00%	1.00%	1.00%	0.99%
Probe	0.99%	0.99%	0.99%	0.99%	0.99%
R2L	0.62%	0.45%	0.53%	0.45%	0.99%
U2R	0.99%	0.89%	0.93%	0.89%	0.99%

Fig. 7. Multi-class classification results on KDD Test+.



	Accuracy	precision	recall/DR	F1-score
micro-average	99.31%	99.32%	99.29%	99.30%
macro-average	99.31%	92%	87%	89%
weighted-average	99%	99.31%	99.29%	99.29%

Fig. 8. Overall proposed results for multi-class classification.

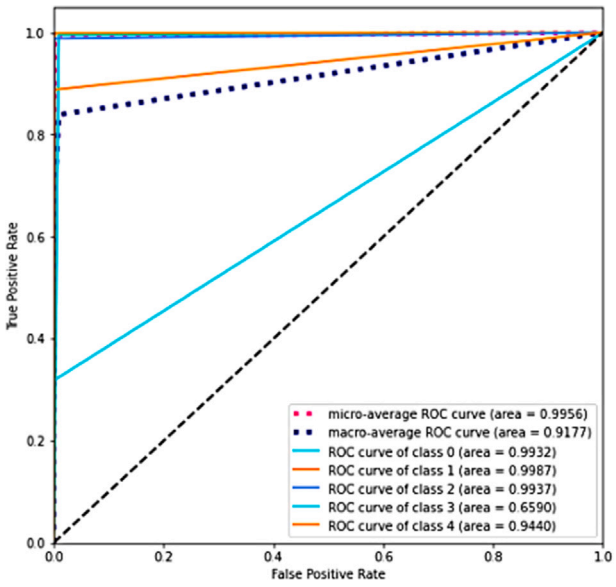


Fig. 9. ROC Curve for multi-class classification.

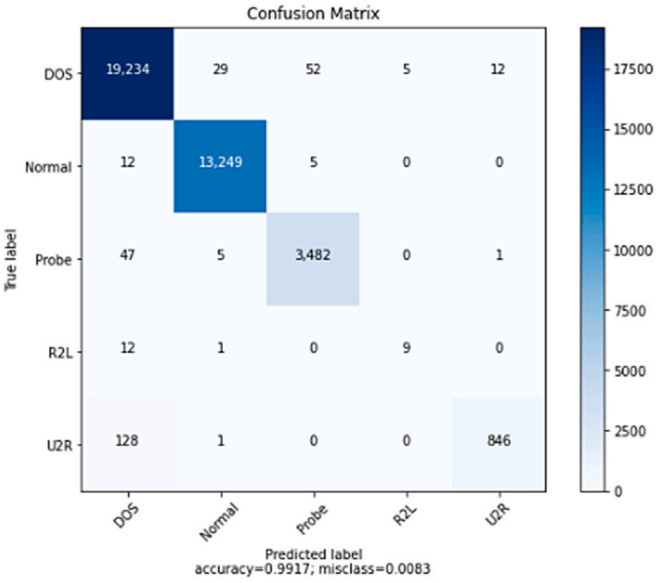


Fig. 10. Confusion Matrix for multi-class.

we can see that the Validation loss is slightly downward according to the Training loss. During epoch 5, the validation and training losses are both 0.10%. The validation and training losses decrease with each epoch iteration, indicating that the model is not over-fitting. The validation and training losses are the same for the 50 iteration epoch, demonstrating that the model performs well on the NSL-KDD dataset.

Fig. 7 shows the KDD Test+ results for five class labels. . In that case, the R2L attack precision is 62%, F1-score is 53% recall, and DR is 0.45%. The overall accuracy is very high on multi-class classifications. The model's performance depends upon the hyper-parameter setting, such as learning rate, epoch, and batch size. Moreover, on support samples in every attack.

Fig. 7 illustrates each label class of normal, Dos, Probe, R2L and U2R on the suggested evaluation matrix. Fig. 8 represents the overall outcome of the proposed model for multi-class. The proposed model achieved 99.31% accuracy for the NSL-KDD dataset. The model calculates macro-average, micro-average, and weighted-average results for precision, recall, and F1-score, ID.

Fig. 9 shows the ROC (receiver operating characteristics curve) for all the classes. This curve is used for classification problems for multi-class classification under various threshold settings. The x-axis represents the FPR, and the y-axis represents the TPR. The multiple-layer ROC curve Since all of the labels' class lines are located at the top of the ROC curve, the DeepDetect methodology fits quite well. The AUC values of class 0 (DoS attack), class 1 (Normal attack), class 2 (probe attack), and class 3 (U2R attack) value is very high, which is greater than 94%. It implies that the model positively impacts these four types of multi-class classification. The AUC value of class 4 (R2L) is 65.9%, which is high compared to the other approaches. The model performed well overall, with a micro-average ROC value of 99.56%.

Fig. 10 illustrates the confusion matrix of multi-class classifications across test sets. The y-axis represents true labels, whereas the x-axis represents predicted labels. The overall accuracy is 99.31%. Each class and output are denoted by "n", whereas multi-class classification problems get a 5*5 confusion matrix. There are four terms to know in the confusion matrix: TP, TN, FP, and FN, according to which we can calculate the accuracy for each class label. The overall accuracy on the test set is high on 4 label classes such as Dos, Normal, Probe, and U2R, as these attacks have high supporting samples; because of that, the weight given to these attacks is too much.

6.2. Results of binary-class classification

The detailed description of binary-class classification results is described in this subsection. The model uses a binary entropy of 0.00007

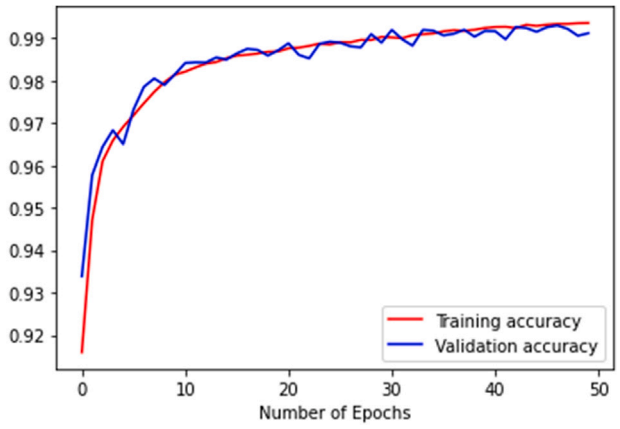


Fig. 11. Training and Validation Accuracy.

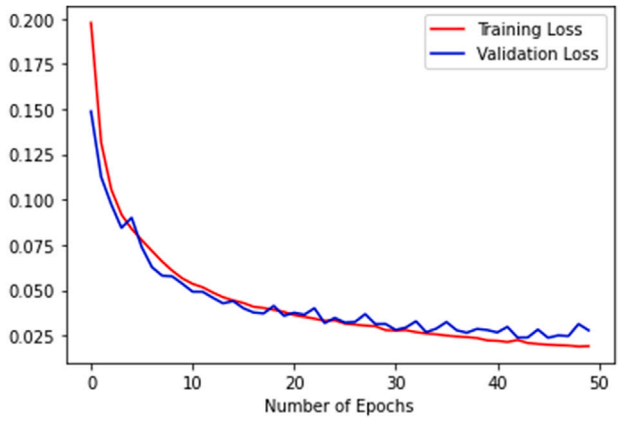


Fig. 12. Training and Validation loss.

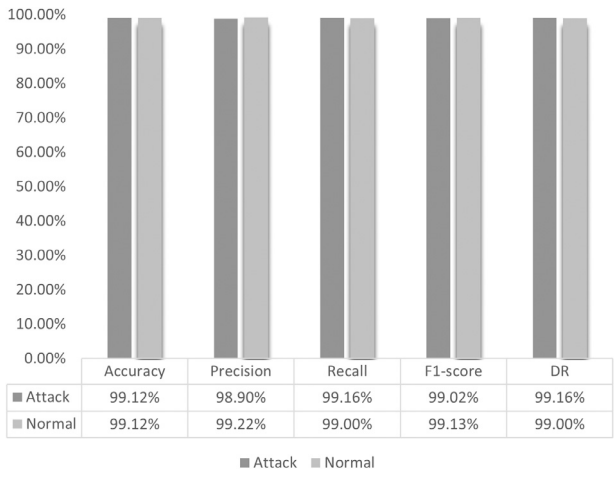


Fig. 13. Binary-class classification results.

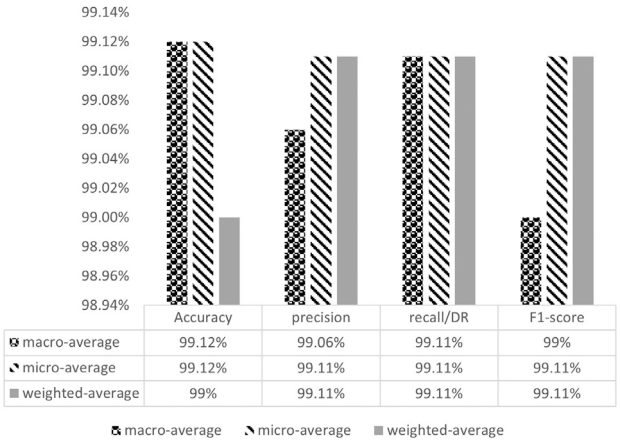


Fig. 14. Overall proposed results for binary-class.

as the learning rate for binary classifications. The proposed model epoch for the train is 50 iterations. Furthermore, all layers have the "same" padding. The sigmoid activation function is set for binary classifications with batch size 30 at the output layers. Fig. 11 depicts the training accuracy and validation graph. The execution time taken by the proposed hybrid DL model for multi-class classification on the NSL-KDD dataset is 1221.573 s, 1221573.06 ms, and 20.3 min.

The validation accuracy has gradually increased concerning the training accuracy, resulting in better accuracy on the training set. The number of epochs must be specified on the x-axis, and the training and validation accuracy should be set on the y-axis. The training and validation accuracy change with each epoch, as shown by the fact that the validation and training accuracy is 97% after 10 epochs. Validation and training accuracy are increasing with each epoch, which is a good sign that the model is not over-fitting or under-fitting because both validation and training accuracy are gaining strength upward. We set 50 epoch values; the epoch ends up with 50 iterations. Fig. 12 depicts the Validation and Training losses. The Validation loss is slightly lower than the Training loss, as seen here. At epoch 6, the validation and training losses are both 0.10%. The validation loss and training loss decrease with each epoch iteration, which is a good sign that the model is not overfitting or underfitting. The validation and training losses are the same for 50 epoch iterations, 0.05, indicating that the model performs well on the dataset. Thus results of the KDD Test+ for two-class labels, normal and attack, are shown in Fig. 13. It shows that the performance of all evaluation matrices is very high for each label class. Because the support samples for these labels are large, the model has given too many weights to attack and normal. The overall performance of the suggested model for the binary class is depicted in Fig. 14. The ROC for the normal and attack label classes is shown in Fig. 15. This curve was used to solve binary classification problems with various threshold settings. The x-axis represents the FPR, while the y-axis represents the TPR. The ROC curve of the DeepDetect technique matches quite well since all of the labels' class lines are at the top of the ROC curve. The AUC values for class 1 (Attack) and class 0 (Normal) are both very high, exceeding 99%. It means the model has a good effect on binary-class classification. We used a confusion matrix to see if the model could accurately predict test data after training. The confusion matrix of binary class classifications over test sets is depicted in Fig. 15. The y-axis represents true labels, whereas the x-axis indicates predicted labels. Overall, the accuracy is 99.12%. For binary class classification problems, each class and output are designated by "n", resulting in a 2*2 confusion matrix. The confusion matrix has four terms to be aware of: TP, TN, FP, and FN. We can calculate the accuracy of each class label based on this. Because the supporting samples are large, the overall accuracy of the test set is high for both the normal and attack classes. As a result, the weights given to these attacks are excessive. Fig. 13 illustrates each label class of normal and attack on the suggested evaluation matrix. The overall performance of the suggested model for the binary class is depicted in Fig. 16. The TP value in the binary-class confusion matrix is 19,153, the FN value is 179, the TN value is 17,649, and the FP value is 149. Thus, the overall performance of the proposed model was calculated using a confusion matrix. The model calculates macro-average, micro-average, and weighted-average results for precision, recall, and f1-score.

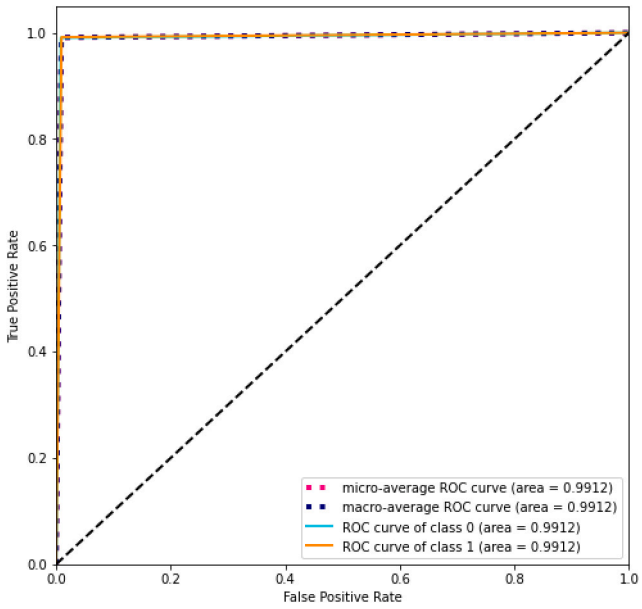


Fig. 15. ROC Curve for binary classification.

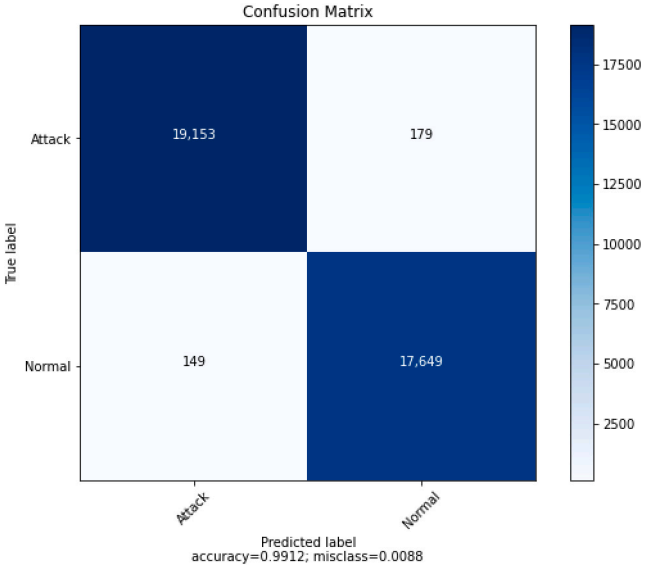


Fig. 16. Confusion Matrix for binary-class.

7. Comparison with the SOTA model for NSL-KDD

The suggested model is compared with the baseline model, such as AE, LCV-AE, CNN, RNN, and MS-DHPN. The proposed model evaluates all model performance by evaluation metrics. Comparison with the SOTA Model for binary-class classification is discussed in Section 7.1, and comparison with the SOTA Model for multi-class classification is discussed in Section 7.2.

7.1. Comparison with the SOTA model for binary-class classification

A comparison of the proposed model with some related work on binary-class classification is shown in Table 5. The author proposed AE and LCV-AE models for anomaly detection. In Table 4, the accuracy of the AE model is 90.61%, LCV-AE model is 85.5% which is low compared to the proposed model. Furthermore, the FPR of the AE model is 17.10%, which is too high compared to the proposed model,

Table 4
Comparison with SOTA model for binary-class.

Model	Accuracy	FPR
AE	90.61%	17.10%
LCV-AE	85.51%	–
RNN	83.28%	3.06%
MR-DHPN	85.9%	–
BC+KNN	94.92%	1.59%
SAAE-DNN	87.74%	–
Google LeNet	81.84%	–
SDAE-ELML	78.04%	–
CNN+GRU	93.10%	1.12
GRU-DAE	90.56%	–
CNN-LSTM	82.6%	–
Proposed Model	**99.12%**	**0.8%**

Table 5
Comparison of the proposed model with SOTA model for multi-class.

Model	Accuracy	FPR
MR-DHPN	80.2%	–
RNN	81.29%	6.50%
CNN	79.48%	7.18%
SAAE	82.14%	–
DSSTE+AlexNet	82.84%	–
RUS+LSTM	77.5%	–
ROS+AlexNet	78.5%	–
Proposed Model	**99.31%**	**0.2%**

which is 0.8%. Likewise, the authors proposed an RNN model for anomaly detection. RNN model gives reliable accuracy in binary-class classification compared to ML approaches such as j48, RF, and NB. However, the RNN model has a problem of exploiting and vanishing gradient. Moreover, when the accuracy of RNN is compared to the proposed model, RNN accuracy is low, which is 83.28%, and FPR is too high, which is 3.06%. The research has also used hybrid models such as BC+KNN, which has high accuracy that is 94.92% as compared to the other hybrid models such as SAAE-DNN, GoogleLeNet, SDAE-ELML, CNN-GRU, SAAE-DNN, GRU-DAE, MR-DHPN and CNN-LSTM. The proposed model achieved a low FPR that is 0.8% and 99.12% accuracy as compared to the baseline method. According to this table, the proposed model outperforms all SOTA models.

7.2. Comparison with the SOTA model for multi-class classification

Table 5 illustrates that the suggested model outperforms all SOTA approaches. For anomaly detection, the authors proposed a hybrid DL model called MR-DHPN. However, the accuracy of the MR-DHPN model is slightly low and ignores the detection of FPR. Likewise, researchers have suggested RNN and CNN models for anomaly detection. However, the RNN model has a problem of exploiting and vanishing gradient. Moreover, when the accuracy of RNN is compared to the proposed model, the RNN accuracy is low, which is 81.29%. A single CNN model cannot fully exploit the rich information in network traffic. Moreover, The accuracy of the CNN model is low compared to the suggested DeepDetect architecture. The proposed model achieved low FPR, which is 0.2% and 99.31% accuracy, compared to the baseline method. It demonstrates that the suggested model outperforms all SOTA models in multi-class classification. Similarly, the suggested model is evaluated against SAAE, DSSTE+AlexNet, RUS+LSTM, and ROS+AlexNet.

8. Conclusion

In Artificial intelligence (AI), DL gained research interest. The Hybrid DL model is used for network anomaly detection both for binary and multiple-class classification. This study proposed a novel approach for network anomaly called DeepDetect, which consists of multiple

layers of CNN, GRU, and Bi-LSTM. We use several convolution neural networks to extract spatial information from network data to understand spatial features. Furthermore, gate recurrent units successfully capture the correlation between the features while overcoming the problem of gradient disappearance. A Bi-LSTM technique was also applied. This layer benefits from the long-term preservation of historical context and extraction of temporal information from backward and forward network traffic data. The suggested approach not only has a high DR, but it also has a strong displaying ability for identification in binary and multi-class classification. In comparison to the SOTA DL classification model, this is especially true that the proposed approach has high DR for both multi and binary class classifications on the NSL-KDD benchmark data set. The proposed model outperforms other methods, as evidenced by the experimental results. The overall accuracy for multi-class classification is 99.31%, and for binary-class classification, it is 99.12%.

The following are some research directions and drawbacks for the researchers.

1. The NSL-KDD dataset is imbalanced, with some attack types being much less common than others. This disparity may skew the evaluation of intrusion detection systems' performance, perhaps resulting in inflated accuracy numbers. To solve this issue We will extend overwork to apply data augmentation techniques to imbalanced classes in ID.
2. Extend this framework to other domains, such as botnet and malware detection.
3. Train and test the proposed model on any real-time IDS.
4. As the proposed model works well on the big dataset in future work, we will check the performance on a small dataset.

CRediT authorship contribution statement

Zeenat Zulfiqar: Writing – original draft, Methodology. **Saif U.R. Malik:** Supervision, Methodology, Investigation, Conceptualization. **Syed Atif Moqurrab:** Writing – review & editing, Validation, Project administration, Formal analysis. **Zubair Zulfiqar:** Visualization, Validation, Resources, Formal analysis, Data curation, Conceptualization. **Usman Yaseen:** Writing – review & editing, Visualization, Validation, Supervision, Methodology. **Gautam Srivastava:** Writing – review & editing, Supervision, Resources, Project administration, Investigation, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research has been supported by the Estonian Research Council grant PRG1780.

References

[1] N. Tariq, M. Asim, Z. Maamar, M.Z. Farooqi, N. Faci, T. Baker, A mobile code-driven trust mechanism for detecting internal attacks in sensor node-powered IoT, J. Parallel Distrib. Comput. 134 (2019) 198–206.
[2] M.W. Akhtar, S.A. Hassan, R. Ghaffar, H. Jung, S. Garg, M.S. Hossain, The shift to 6G communications: vision and requirements, Hum.-Cent. Comput. Inf. Sci. 10 (1) (2020) 1–27.
[3] M. Wang, T. Zhu, T. Zhang, J. Zhang, S. Yu, W. Zhou, Security and privacy in 6G networks: New areas and new challenges, Digit. Commun. Netw. 6 (3) (2020) 281–291.
[4] M. Almiani, A. AbuGhazleh, A. Al-Rahayfeh, S. Atiewi, A. Razaque, Deep recurrent neural network for IoT intrusion detection system, Simul. Model. Pract. Theory 101 (2020) 102031.
[5] F.A.M. Khiralla, Statistics of cybercrime from 2016 to the first half of 2020.
[6] W. Xu, J. Jang-Jaccard, A. Singh, Y. Wei, F. Sabrina, Improving performance of autoencoder-based network anomaly detection on nsl-kdd dataset, IEEE Access 9 (2021) 140136–140146.
[7] F.A. Khan, A. Gumaei, A. Derhab, A. Hussain, A novel two-stage deep learning model for efficient network intrusion detection, IEEE Access 7 (2019) 30373–30385.
[8] I. Ullah, Q.H. Mahmoud, Design and development of a deep learning-based model for anomaly detection in IoT networks, IEEE Access 9 (2021) 103906–103926.
[9] I. Apostol, M. Preda, C. Nila, I. Bica, IoT botnet anomaly detection using unsupervised deep learning, Electronics 10 (16) (2021) 1876.
[10] K. Wu, Z. Chen, W. Li, A novel intrusion detection model for a massive network using convolutional neural networks, IEEE Access 6 (2018) 50850–50859.
[11] J. Gu, L. Wang, H. Wang, S. Wang, A novel approach to intrusion detection using SVM ensemble with feature augmentation, Comput. Secur. 86 (2019) 53–62.
[12] G. Andresini, A. Appice, D. Malerba, Nearest cluster-based intrusion detection through convolutional neural networks, Knowl.-Based Syst. 216 (2021) 106798.
[13] C. Yin, Y. Zhu, J. Fei, X. He, A deep learning approach for intrusion detection using recurrent neural networks, IEEE Access 5 (2017) 21954–21961.
[14] X. Xu, J. Li, Y. Yang, F. Shen, Toward effective intrusion detection using log-cosh conditional variational autoencoder, IEEE Internet Things J. 8 (8) (2021) 6187–6196.
[15] M. Hasan, M.M. Islam, M.I.I. Zarif, M. Hashem, Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches, Internet Things 7 (2019) 100059.
[16] C.S. Rao, K.B. Raju, Mapreduce accelerated signature-based intrusion detection mechanism (idm) with pattern matching mechanism, in: Soft Computing in Data Analytics, Springer, 2019, pp. 157–164.
[17] L. Li, Y. Yu, S. Bai, Y. Hou, X. Chen, An effective two-step intrusion detection approach based on binary classification and k -NN, IEEE Access 6 (2018) 12060–12073.
[18] I. Ahmad, M. Basheri, M.J. Iqbal, A. Rahim, Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection, IEEE Access 6 (2018) 33789–33795.
[19] X. Gao, C. Shan, C. Hu, Z. Niu, Z. Liu, An adaptive ensemble machine learning model for intrusion detection, IEEE Access 7 (2019) 82512–82521.
[20] S.U. Jan, S. Ahmed, V. Shakhov, I. Koo, Toward a lightweight intrusion detection system for the internet of things, IEEE Access 7 (2019) 42450–42471.
[21] M.S. Elsayed, N.-A. Le-Khac, S. Dev, A.D. Jurcut, Machine-learning techniques for detecting attacks in SDN, in: 2019 IEEE 7th International Conference on Computer Science and Network Technology, ICCSNT, IEEE, 2019, pp. 277–281.
[22] M.S. Ansari, V. Bartoš, B. Lee, GRU-based deep learning approach for network intrusion alert prediction, Future Gener. Comput. Syst. 128 (2022) 235–247.
[23] Y. Xiao, C. Xing, T. Zhang, Z. Zhao, An intrusion detection model based on feature reduction and convolutional neural networks, IEEE Access 7 (2019) 42210–42219.
[24] X. Wang, S. Yin, H. Li, J. Wang, L. Teng, A network intrusion detection method based on deep multi-scale convolutional neural network, Int. J. Wirel. Inf. Netw. 27 (4) (2020) 503–517.
[25] S. Kim, A. Jing, H. Park, G.A. Lee, W. Huang, M. Billinghurst, Hand-in-air (hia) and hand-on-target (hot) style gesture cues for mixed reality collaboration, IEEE Access 8 (2020) 224145–224161.
[26] K. Thapa, N. Duraipandian, Malicious traffic classification using long short-term memory (LSTM) model, Wirel. Pers. Commun. 119 (3) (2021) 2707–2724.
[27] Y. Imrana, Y. Xiang, L. Ali, Z. Abdul-Rauf, A bidirectional LSTM deep learning approach for intrusion detection, Expert Syst. Appl. 185 (2021) 115524.
[28] H. He, X. Sun, H. He, G. Zhao, L. He, J. Ren, A novel multimodal-sequential approach based on multi-view features for network intrusion detection, IEEE Access 7 (2019) 183207–183221.
[29] C. Tang, N. Luktarhan, Y. Zhao, SAAE-DNN: Deep learning method on intrusion detection, Symmetry 12 (10) (2020) 1695.
[30] A. Chen, Y. Fu, X. Zheng, G. Lu, An efficient network behavior anomaly detection using a hybrid DBN-lstm network, Comput. Secur. 114 (2022) 102600.
[31] M. Turkoz, S. Kim, Y. Son, M.K. Jeong, E.A. Elsayed, Generalized support vector data description for anomaly detection, Pattern Recognit. 100 (2020) 107119.
[32] Q. Ding, J. Li, Anogla: An efficient scheme to improve network anomaly detection, J. Inf. Secur. Appl. 66 (2022) 103149.
[33] R.K. Malaiya, D. Kwon, S.C. Suh, H. Kim, I. Kim, J. Kim, An empirical evaluation of deep learning for network anomaly detection, IEEE Access 7 (2019) 140806–140817, http://dx.doi.org/10.1109/ACCESS.2019.2943249.
[34] S. Naseer, Y. Saleem, S. Khalid, M.K. Bashir, J. Han, M.M. Iqbal, K. Han, Enhanced network anomaly detection based on deep neural networks, IEEE Access 6 (2018) 48231–48246.
[35] Z. Chen, C.K. Yeo, B.S. Lee, C.T. Lau, Autoencoder-based network anomaly detection, in: 2018 Wireless Telecommunications Symposium, WTS, IEEE, 2018, pp. 1–5.
[36] M. Imran, N. Haider, M. Shoaib, I. Razzak, et al., An intelligent and efficient network intrusion detection system using deep learning, Comput. Electr. Eng. 99 (2022) 107764.
[37] L. Dhanabal, S. Shantharajah, A study on NSL-kdd dataset for intrusion detection system based on classification algorithms, Int. J. Adv. Res. Comput. Commun. Eng. 4 (6) (2015) 446–452.